OFDM을 위한 Multipath Delay Feedback (MDF) FFT 프로세서 설계

정회원 이 종 민*, 정 용 진**

Design of a Multipath Delay Feedback(MDF) FFT processor for OFDM

Jong-Min Lee*, Yong-Jin Jeong** *Regular Members*

요 약

OFDM(Orthogonal Frequency Division Multiplexing)은 제4세대 기술로 일컬어지는 변조 방식으로 최근 유럽의 디지털 오디오 방송(DAB)과 디지털 비디오 방송(DVB)에 표준으로 사용되고 있으며, IEEE 802.11a 무선 LAN 및 디지털 가입자 라인 xDSL 에서도 사용되고 있다. 본 논문에서는 OFDM 모뎀 구현의 핵심이라고 할 수 있는 64-포인트 FFT(Fast Fourier Transform) 프로세서의 여러 가지 구조를 분석하고, 이들과 비교하여 성능 대 면적 비를 획기적으로 향상시킨 새로운 FFT 프로세서인 Radix-2 MDF (Multipath Delay Feedback) 구조를 제안하였다. 본 논문에서 제안하는 MDF 구조는 현재 가장 널리 사용되고 있는 Radix-2 SDF (Singlepath Delay Feedback)[1] 구조에 비해 복소 곱셈기의 수를 절반으로 줄여 성능의 저하 없이 하드웨어의 크기를 획기적으로 줄인 결과를 보여준다. FFT 프로세서에서 복소 곱셈기가 전체 면적의 약 80% 를 차지하는 것을 감안하면 SDF 방식에 비해 면적이 60% 정도로 줄어든 효과를 주게 된다. 본 논문의 제안 방식은 1024-포인트 혹은 그 이상의 FFT 크기를 갖는 무선 LAN, xDSL, DAB, DVB 등에도 동일하게 적용 가능하며 현재 국내외에 발표된 논문 중 성능 대 면적비가 가장 우수한 구조로 여겨진다.

ABSTRACT

OFDM(Orthogonal Frequency Division Multiplexing) is a next generation technique for data modulation. It has been adopted as a standard in DAB and DVB in Europe, and also in IEEE 802.11a WLAN and digital subscriber line(xDSL) systems. In this paper, we analyze previous approaches of a 64 point FFT(Fast Fourier Transform) processor, which is a core module of the OFDM modem, and propose a new FFT architecture, what we call Radix-2 MDF(Multipath Delay Feedback), that can greatly reduce hardware area without any performance degradation. Compared to the most widely used architecture, Radix-2 SDF, the MDF uses only half number of complex multipliers. Considering that complex multipliers occupy about 80% of the entire FFT processor, the MDF requires only 60% hardware area of the SDF. The Radix-2 MDF can also be applied to much bigger FFT processors, such as 1024 or 8048-point FFT, with the same degree of area advantage. To our knowledge, this can be considered as one of the most efficient architecture to implement a FFT processor for high performance VLSI or ASIC.

* 광운대학교 전자통신공학과 실시간구조 연구실(sonnet@explore.gwu.ac.kr)
** 광운대학교 전자공학부 조교수 (yjjeong@daisy.gwu.ac.kr)
논문번호 : 020264-0605, 접수일자 : 2002년 6월 5일
※ 본 연구는 광운대학교 반도체 설계 교육 센터(IDEC) 및 2002 광운대학교 산학연 센타의 지원으로 연구되었습니다.

Ⅰ. 서 론

OFDM(Orthogonal Frequency Division Multiplexing)의 기본원리는 여러개의 반송파를 사용하여 고속 전송률을 갖는 입력 데이터를 낮은 전송률을 갖는 데이터로 반송파의 수만큼 병렬화 하여 각 반송파에 실어 전송하는 방식이다. 이때, 낮은 전송률을 갖는 부반송파의 심벌 듀레이션이 증가하기 때문에 다중경로 지연확산에 의한 시간상에서의 상대적인 왜곡이 감소하고, 모든 OFDM 심벌 사이에 채널의 지연확산보다 긴 보호구간을 삽입하여 심벌간 간섭을 제거할 수 있다.[2]

이러한 OFDM 변복조는 다수의 부반송파를 사용하기 때문에 부반송파의 수가 많아지면 하드웨어 설계에 대한 어려움이 매우 크다. 또한 부반송파간의 직교성을 유지하기 위한 어려움이 발생하여 실제 시스템에 구현하기가 어려워진다. 이러한 문제점을 DFT(Discrete Fourier Transform)에 의해 구현할 수 있으며 DFT의 많은 연산량을 줄이기 위해 FFT 알고리즘을 이용하여 구현한다.[2]

FFT 알고리즘은 시간 데시메이션 또는 주파수 데시메이션에서 길이가 N인 DFT를 보다 작은 길이의 DFT로 연속해서 분해하여 계산하는 것이다. 이렇게 함으로써 DFT의 연산량에서 곱셈을 N^2번에서 NlogN번으로 줄일 수 있게 되는데,[2] 이러한 FFT의 효율적인 연산을 위해서 가장 많이 사용되는 알고리즘이 Radix-2 DIT(Decimation In Time) FFT와 DIF(Decimation In Frequency) FFT, Radix-4 DIT FFT와 DIF FFT이다.[3]

OFDM에서의 FFT는 실수부와 허수부를 갖는 복소수 연산을 수행한다. 따라서 하드웨어적으로는 실수부와 허수부가 분리되어 입력되어지는데, FFT 프로세서를 설계하여 입출력의 실수부와 허수부의 위치를 바꾸면 Inverse FFT 역할을 수행할 수 있다.[4] 이러한 FFT 알고리즘의 구현은 크게 어레이 방식과 파이프라인 방식이 있다. 어레이 FFT 구조는 하드웨어적으로 매우 복잡하고 커져서 FFT의 연산점(N)이 큰 경우(256,2k,8k..etc)에 구현이 거의 불가능하다. 반면에 파이프라인 FFT 구조는 규칙적이고 비교적 제어가 간단하며 직렬 입력과 직렬 출력을 할 수 있기 때문에 높은 성능을 요구하는 응용분야에 가장 많이 사용하는 구조이다. 여러 가지 파이프라인 방식의 FFT중에서 본 논문에서는 Radix-2 SDF 방식에서 복소곱셈기의 수를 절반으로 감소시킬 수 있는 새로운 FFT 구조를 제안하고 이를 이용해 64 포인트 FFT를 설계 검증하였다.

Ⅱ. FFT 알고리즘

2.1 FFT의 개념

주기 신호의 Fourier Transform은 식(1)과 같이 정의한다.[3]

$$X(f)=\int_{-\infty}^{\infty}x(t)\exp(-j2\pi ft)dt$$

$$x(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty}X(f)\exp(j2\pi ft)df$$

$$(\text{단, } j=\sqrt{-1}) \quad (1)$$

그러나 실제 응용에서 사용할 수 있는 시간영역이나, 주파수영역의 신호는 이산적으로 분포하게 되는데, 이 이산 신호 분포에 따른 Fourier Transform 즉, DFT는 다음과 같이 정의한다.[3]

$$X(k)=\sum_{n=0}^{N-1}x(n)\exp[-j\frac{2\pi nk}{N}]$$

$$x(n)=\frac{1}{N}\sum_{n=0}^{N-1}X(k)\exp[j\frac{2\pi nk}{N}] \quad (2)$$

(단, N은 sample의 개수, k = 0, 1, 2, ..., N-1)

식(2)에서 $\exp[-j\frac{2\pi nk}{N}]=W_N^{nk}$ 라고 할 때, 다음과 같이 정리 할 수 있다.[3]

$$X(k)=\sum_{n=0}^{N-1}x(n)\,W_N^{nk},$$

$$=\sum_{n=0}^{N-1}[Re\{x(n)\}Re\{W_N^{nk}\}-Im\{x(n)\}Im\{W_N^{nk}\}$$

$$+j(Re\{x(n)\}Im\{W_N^{nk}\}+Im\{x(n)\}Re\{W_N^{nk}\}]$$

$$(\text{단, } k=0,\ 1,\ 2,\ 3,\ \cdots,\ N-1) \quad (3)$$

DFT는 식(3)에서 보듯이 복소곱셈이 필요하며, N값이 커질수록 DFT의 계산량은 N^2에 비례하여 증가한다. 표(1)은 DFT의 계산량을 보였다.

표 1. DFT의 계산량

연산 / 종류	Multiply	Add
Real	$4N^2$	$N\,(\,4N-2\,)$
Complex	N^2	$N\,(\,N-1\,)$

1965년에 Cooley와 Tukey[3]는 DFT 연산 중에 수행할 계산의 양을 실질적으로 줄이는 알고리즘을

개발하였다. 이러한 효율적인 알고리즘을 FFT라고 한다. 이 알고리즘으로 인하여 디지털 신호 처리 분야를 포함하여 여러 분야에서 DFT를 많이 사용하게 되었다. 그리고 다른 효율적인 알고리즘이 개발되는 계기가 되었다. 본 논문에서는 새로운 FFT 프로세서 구조를 설계함에 있어서 Cooley와 Tukey의 알고리즘에서 주파수 데시메이션에서의 알고리즘 Radix-2 DIF FFT를 사용하였다.

2.2 Radix-2 DIF 알고리즘

Radix-2 DIF FFT 수식을 정리해 보면, N-포인트 DFT의 식은 FFT 알고리즘의 형태로 변형되어 다음 과정과 같이 된다.

$$X(k)=\sum_{n=0}^{N-1}x(n)\,W_N^{kn}$$
$$=\sum_{n=0}^{N/2-1}x(n)\,W_N^{kn}+\sum_{n=N/2}^{N-1}x(n)\,W_N^{kn}$$
$$=\sum_{n=0}^{N/2-1}x(n)\,W_N^{kn}+$$
$$\sum_{n=0}^{N/2-1}x(N/2+n)\,W_N^{k(N/2+n)}=\sum_{n=0}^{N/2-1}x(n)\,W_N^{kn}+$$
$$W_N^{kN/2}\sum_{n=0}^{N/2-1}x(N/2+n)\,W_N^{kn} \qquad (4)$$

식(4)에서 $W_N^{kN/2}$ 를 다른 값으로 변환이 가능한데, 그 식은 식(5)와 같다. 또한 식(5)를 식(4)에 대입하면 식(6)이 된다.[3]

$$W_N^{kN/2}=e^{(-j2\pi k/N)\times N/2}=(-1)^k \qquad (5)$$

$$X(k)=\sum_{n=0}^{N/2-1}x(n)\,W_N^{kn}+$$
$$(-1)^k\sum_{n=0}^{N/2-1}x(N/2+n)\,W_N^{kn} \qquad (6)$$

먼저 k가 짝수인 경우에 식(6)에 k=2r을 대입하면 식(7)와 같이 정리할 수 있다. 그리고 k가 홀수인 경우에는 k=2r+1로 나타낼 수 있으므로, 짝수인 경우와 마찬가지로 식(6)에 k=2r+1을 대입하면 식(8)로 정리된다. k가 짝수인 경우와 홀수인 경우에 대한 것을 하나로 묶어서 정리하면 식(9)와 같되고, 식(9)를 그림으로 표현하면 그림(1)과 같다.

$$X(2r)=\sum_{n=0}^{N/2-1}x(n)\,W_N^{2rn}+\sum_{n=0}^{N/2-1}x(N/2+n)\,W_N^{2rn}$$
$$=\sum_{n=0}^{N/2-1}[x(n)+x(N/2+n)]\,W_N^{2rn}$$
$$=\sum_{n=0}^{N/2-1}[x(n)+x(N/2+n)]\,W_{N/2}^{rn} \qquad (7)$$

$$X(2r+1)=\sum_{n=0}^{N/2-1}x(n)\,W_N^{2rn+n}-$$
$$\sum_{n=0}^{N/2-1}x(N/2+n)\,W_N^{2rn+n}$$
$$=\sum_{n=0}^{N/2-1}[x(n)-x(N/2+n)]\,W_N^{n}W_{N/2}^{rn} \qquad (8)$$

$$X(2r)=\sum_{n=0}^{N/2-1}[x(n)+x(N/2+n)]\,W_{N/2}^{rn}$$
$$X(2r+1)=\sum_{n=0}^{N/2-1}[x(n)-x(N/2+n)]\,W_N^{n}W_{N/2}^{rn} \qquad (9)$$

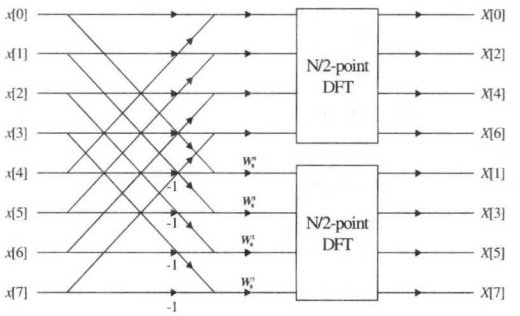


그림 1. 8포인트DFT의 연산 흐름도

두 포인트 FFT의 연산을 그림(2)에 나타내었다.

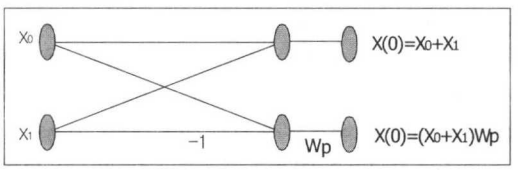


그림 2. 두포인트 FFT 연산

마찬가지 방법으로 N/4, N/8,...., 2 포인트 DFT로 연속해서 분리하면 8포인트 FFT의 최종적인 연산 흐름도는 그림(3)과 같다.

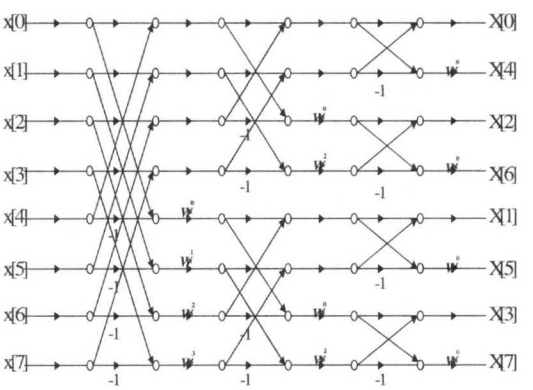


그림 3. Radix-2 DIF FFT의 최종 플로우 그래프

입력 데이타열은 정상순서이고 출력 데이타열은 비트역순임을 그림(3)에서 알 수 있다. 즉 위와 같은 경우, 입력이 정상순서이면 출력은 비트역순으로 받아들여야 한다. 비트역순 배열을 나타내는 트리 구조를 그림(4)에 나타내었다.

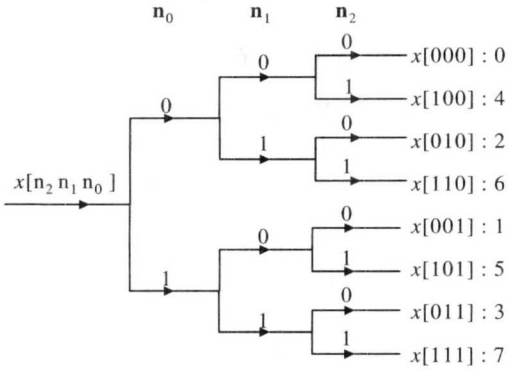


그림 4. 8포인트DIF FFT의 비트역순 트리구조

Ⅲ. Previous Work

파이프라인 방식으로 구현된 여러 가지 FFT의 구현 사례에서 각 스테이지간의 패스가 하나인지 여러개인지, 또한 Radix-2인지 Radix-4인지에 따라서 각각의 구조와 동작을 설명하였다.

• R2 MDC(Multipath Delay Commutator)[5]

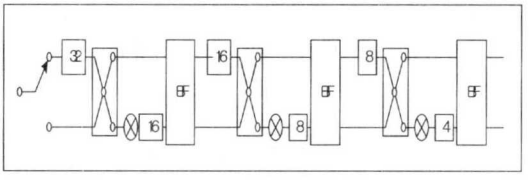


그림 5. Radix 2 MDC FFT 구조(N=64)

각 스테이지 사이에 두 개의 패스가 있다. 입력 데이타열을 두개의 병렬 데이터 열로 나누고 지연소자를 이용하여 두 데이타간의 거리를 조정하여 BF에 입력한다. 지연소자는 이전 스테이지의 1/2이며, BF는 1/2 주기(N/2 점)동안 동작하고 나머지 1/2은 쉬게 된다. 각 스테이지의 스위치는 앞 스테이지 스위치보다 두배 빠르게 스위칭한다.[5]

• R2 SDF(Singlepath Delay Feedback)[1]

각 스테이지 사이에 한 개의 패스가 존재한다. 입력 데이터열을 지연소자에 저장한 후 각 스테이지에 필요한 입력값이 들어올때에 지연소자에 있던

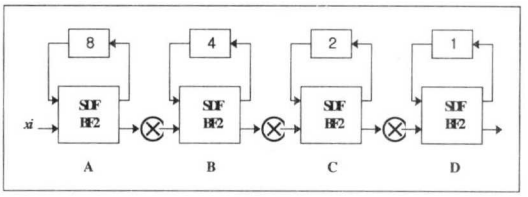


그림 6. Radix 2 SDF FFT 구조(N=16)

값과 BF 연산을 한다. 이때 BF출력의 일부를 피드백하여 비어있는 지연소자에 다시 저장한다. 이와같이 지연소자의 효율을 높임으로써 메모리 크기를 줄이는 방식이다. 이 구조는 BF, 복소곱셈기, 지연소자로 구성되어지며, BF는 N=16인 경우 첫 번째 스테이지에서 8개의 입력을 지연소자에 저장하고 그 다음 입력부터 식(10)을 연산한다.

$$B_i = A_i + A_{i+8}$$
$$B_{i+8} = A_i - A_{i+8} \quad (10)$$

B_i는 두 번째 스테이지로 입력되고, B_{i+8}는 지연소자에 피드백 된다. B_i가 두 번째 스테이지에서 BF연산의 수행을 종료하면 B_{i+8}은 곱셈계수와 곱셈을 수행한다. 이때 두 번째 스테이지에서는 BF연산에 필요한 데이타들이 모두 입력되어 대기 시간 없이 BF 연산을 수행한다.[1]

• R4 MDC(Multipath Delay Commutator)[5]

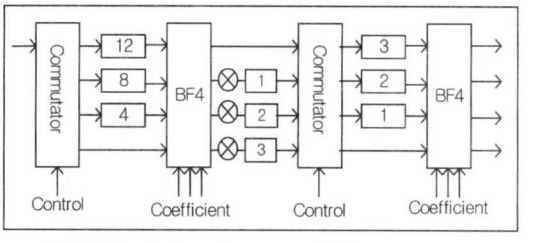


그림 7. Radix 4 MDC FFT 구조(N=16)

Radix-2 MDC 방식의 Radix-4로써 각 스테이지 사이에는 4개의 패스가 있다. 입력 데이터 열을 4개의 병렬 데이터 열로 나누고 지연소자를 이용하여 네 개의 데이타의 거리를 조정하여 BF 에 입력한다. Radix-2 MDC와 마찬가지로 지연소자는 이전 스테이지의 1/2이며, BF는 1/2 주기(N/2 점)동안 동작하고 나머지 1/2는 쉬게 된다. 각 스테이지의 스위치는 이전 스테이지의 스위치보다 두배 빠르게 스위칭을 수행한다.[5]

• R4 SDF(Singlepath Delay Feedback)[6]

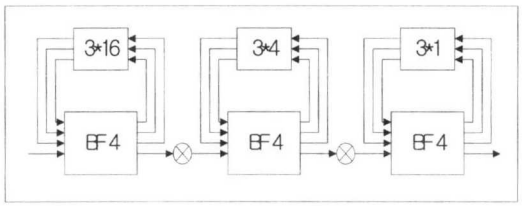


그림 8. Radix 4 SDF FFT 구조(N=64)

Radix-2 SDF방식의 Radix-4 방식으로써, BF 출력 3개의 값을 지연소자에 피드백 함으로써 메모리 크기를 줄인 방식이다.[6]

• R4 SDC(Singlepath Delay Commuatator)[7]

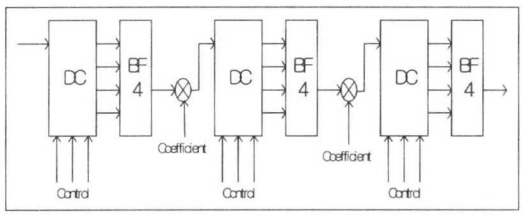


그림 9. Radix 4 SDC FFT 구조(N=64)

각 스테이지가 한 개의 패스로 이루어지고, 각 스테이지에서 BF 입력에 필요한 데이터를 얻도록 DC(Delay Commutator)에 지연소자를 두어 4개의 필요한 데이터를 출력한다. 그리고 BF는 그 4개의 입력을 받아서 1개의 출력만 나온다. 다른 Radix-4 파이프라인 FFT 구조에 비해서 BF의 구조가 간단하지만 많은 수의 지연소자를 사용한다.[7]

• R2^2 SDF(Singlepath Delay Feedback)[8]

Cooley-Tukey 의 알고리즘을 이용하여 복소곱셈의 횟수를 줄이도록 일반적인 Radix-2, Radix-4가 아닌 새로운 방식을 제안한 것이다. 그림(10)에서 보듯이 복소곱셈이 두 스테이지에서 한개 사용된다. 복소곱셈이 이루어지지 않는 스테이지에서는 1/4 점에서만 (-j) 가 곱해진다.

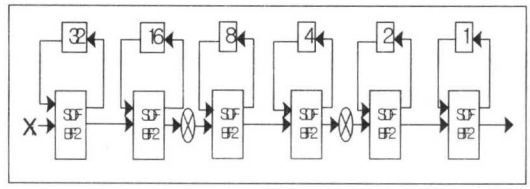


그림 10. Radix-2^2 SDF FFT 구조(N=64)

가장 대표적인 FFT 구현 방식을 알아보았다. 기존의 64포인트 파이프라인 FFT 구조에서 Radix-2에서는 4개의 복소곱셈기를 사용하고, Radix-4에서는 많은 수의 지연소자를 사용하는 대신에 2개의 복소곱셈기를 사용하고 있다. 그러나 본 논문에서는 성능대 면적비로 가장 효율적인 방법인 Radix-2 SDF FFT를 구현하되, 50%의 이용율을 가지는 복소곱셈기를 100%의 이용율을 갖도록 하고 컨트롤을 조절하여 2개의 복소곱셈기만을 사용하는 새로운 구조를 제안하였다. 이렇게 함으로써 OFDM, DVB, DAB 등의 큰 크기의 FFT를 사용하게 되는 경우에 있어서 지연소자를 가장 효율적으로 사용하는 Radix-4 SDF 방식이나, 컨트롤이 조금 복잡한 Radix-4 SDC 방식의 BF 구조가 간단한 방식을 많이 사용하고 있지만, 본 논문의 제안한 구조로서 Radix-2의 간단한 BF 구조와 복소곱셈기 수는 Radix-4와 같으므로 더욱 효율적인 구조라 할 수 있다. 이것은 최근 알고리즘 레벨에서 새로이 제안한 Radix-2^2 SDF 방식과 동일한 하드웨어 리소스를 가지지만 아키텍쳐 레벨에서 변형한 것으로써 보다 간단한 컨트롤을 가진다.

Ⅳ. 새로운 FFT 프로세서 구조의 제안

본 논문에서 제안하는 구조는 SDF 방식을 변형하고 컨트롤을 조절하여 복소곱셈기 수를 절반으로

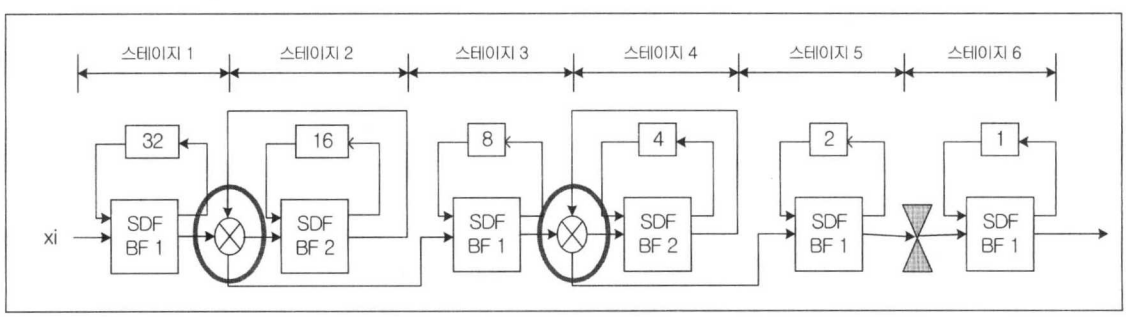


그림 11. 제안한 구조의 FFT 전체 구조(N=64)

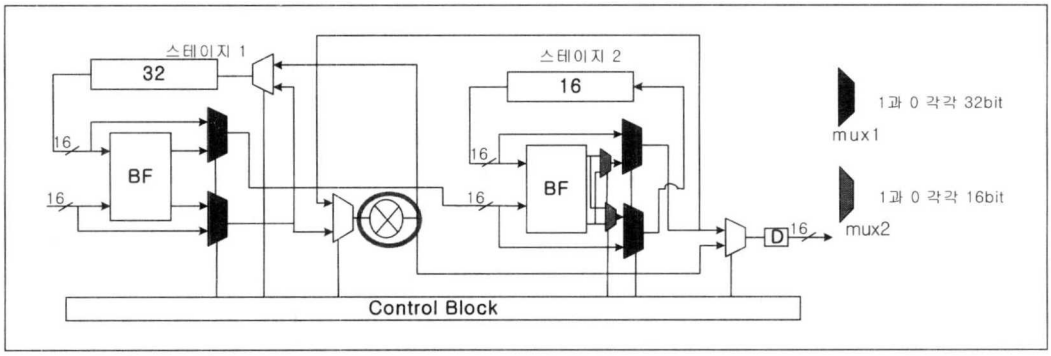


그림 12. 스테이지1,2의 구조

줄인것으로써 두 스테이지에서 한 개의 복소곱셈기를 가지는 구조이다. 전체 블록 다이어그램을 그림(11)에 나타내었다. 스테이지1은 32개의 지연소자를 가지고 있어서 32포인트 떨어진 두 복소수 입력 a와 b에 대해서 식(10)의 BF 연산을 수행한다. 두 입력 a와 b의 a+b는 스테이지2의 입력이 되고, a-b가 된 값은 피드백 되어 지연소자에 다시 저장 된다. 이렇게 하여 a+b가 모두 연산된 후 스테이지2로 입력되면 그때 지연소자에 저장되어 있던 a-b는 복소곱셈기에서 곱셈계수와 연산이 된 후 스테이지2로 입력된다. 이때 복소곱셈기는 a-b에서만 연산이 되기 때문에 그 효율이 50%지만, 본 논문에서는 스테이지2에서의 복소곱셈기에서 연산되는 시간을 조절하여 두 개의 50% 효율을 가지는 복소곱셈기를 한 개의 동일한 복소곱셈기로써 서로 다른 시간에 연산하도록 하여 하나의 복소계산기를 100% 활용할 수 있도록 하였다. 스테이지3과 스테이지4에서도 이와 동일하게 복소곱셈기 사용하는 시간을 조절하여 100%의 이용율을 가지는 하나의 복소곱셈기만을 사용하였다.

스테이지1,2 에서 복소곱셈기 하나를 서로 공유한 구조를 그림(12)에 보였다. 스테이지1에서 a+b와 a-b가 동시에 연산되고, a-b는 곱셈계수와 곱해져서 지연소자로 피드백된다. a-b가 피드백 될 때에 a+b는 스테이지2로 입력되어 연산을 하는데 스테이지2의 구조는 스테이지1의 구조와 다르다. 스테이지1의 연산에서 복소곱셈을 하는 시간에 스테이지2는 복소곱셈을 하지 않는 a+b가 먼저 연산되고, 스테이지1의 a+b가 모두 출력되는 시간이 스테이지1의 복소곱셈이 끝나는 시점이므로, 그 다음부터는 스테이지2의 a-b가 피드백되고 복소곱셈기에 입력되어 연산을 한 후 스테이지3으로 입력된다.

스테이지3,4의 구조는 스테이지1,2의 구조와 동일하다. 스테이지3과 스테이지4 사이에 복소곱셈기 하나가 있어서 스테이지 3에서 복소곱셈을 하고 피드백되었을 때 스테이지 4에서 복소곱셈기를 사용하도록 되어있다. 각 스테이지에는 R2 SDF와는 다르게 mux가 있어서 각각에 컨트롤을 함으로써 두 스테이지가 서로 다른 타이밍에 복소곱셈기를 사용한다.

이렇게 함으로써 R2-SDF 방식보다는 2개의 스테이지에서 mux4개와 레지스터 1개가 더 소모되므로 전체에서 8개의 mux와 2개의 레지스터가 더 사용된다. 반면에 복소곱셈기의 이용율을 50%에서 100%로 최대화 하게 되므로 전체 구조에서 복소곱셈기의 수를 절반으로 줄이는 효과를 가지게 된다.

마지막 스테이지5와 스테이지6의 구조를 그림(13)에 나타내었다. FFT 연산에서 이 구간에 하는 연산은 (-j)만을 곱하는 것이다. 그러므로 복소곱셈기를 설계하는 것이 아니라, 스테이지6으로 입력되는 데이터의 실수값과 복소수값을 스위치하고 복소수 값에 -1을 곱해주도록 하였다. 그림(14)은 스테이지5,6에 있는 스위치의 구조를 나타내었다.

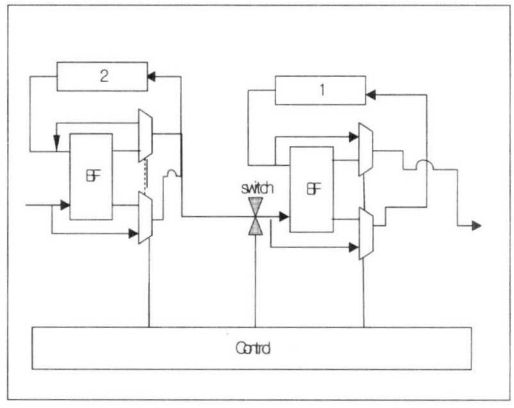


그림 13. 스테이지 5,6의 구조

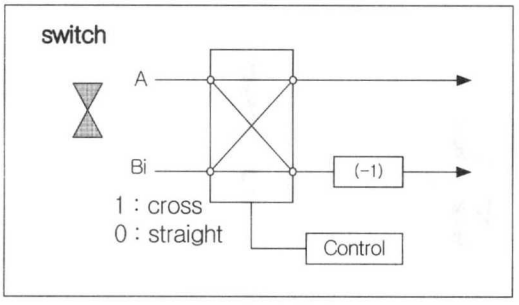


그림 14. 스위치의 구조

• BF(Butterfly)

두 복소수 입력 데이터 $a=A_r+A_i$ 와 $b=B_r+B_i$를 각각 실수부와 허수부로 나누어서 연산을 한다. BF는 그림(15)와 같이 4개의 덧셈기와 4개의 mux로 이루어졌으며, 각각의 출력은 $a_r=(A+B)_r$, $a_i=(A+B)_i$, $b_r=(A-B)_r$, $b_i=(A-B)_i$, 이다.

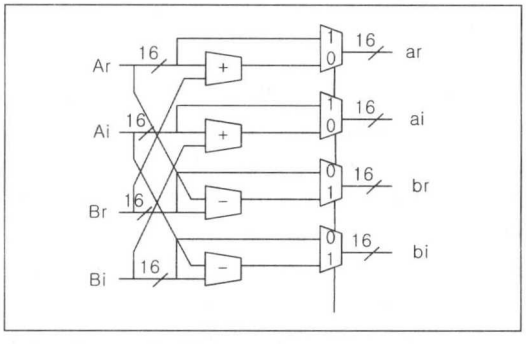


그림 15. Butterfly 구조

• 복소곱셈기

두개의 복소수 입력을 각각 실수부와 허수부로 나누어서 연산한다. 그림(16)과 같이 4개의 곱셈기와 2개의 덧셈기로 이루어졌기 때문에 지연소자, BF, 곱셈계수 테이블등과 함께 이루어진 전체 FFT 구조에서 복소곱셈기는 약 85%의 하드웨어 크기를 점유한다.

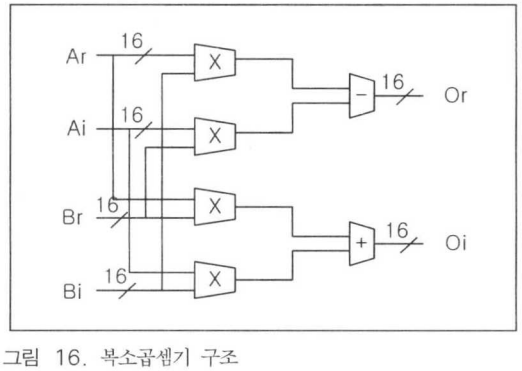


그림 16. 복소곱셈기 구조

• 컨트롤 설계

두 개의 스테이지를 합쳐서 타이밍을 조절하고 복소곱셈기 수를 줄였으므로 두 개의 스테이지 중에서 스테이지2의 컨트롤이 달라진다. 즉 스테이지1에서 복소곱셈기를 사용하지 않는 시간에 스테이지2에서 복소곱셈기를 사용해야 하기 때문이다. R2-SDF에서는 두 복소수 $a= A_r+A_i$와 $b= B_r+B_i$ 가 BF에서 a+b와 a-b를 동시에 연산한 후에 a+b는 스테이지2로 입력되고, a-b는 피드백 되었다가 a+b이 모두 스테이지2로 입력된 후에 복소곱셈을 하고 스테이지2로 입력된다.

그러나 본 논문에서 제안한 방식은 BF에서 a+b는 다음 스테이지로 입력되고 동시에 a-b는 복소곱셈기에서 곱셈계수와 곱해진 후에 피드백 되어 지연소자에 저장된다. 그리고 a+b가 모두 스테이지2로 입력된 후에 a-b도 스테이지2로 입력되는데 이때는 복소곱셈을 하지 않고 스테이지2로 입력되기 때문에 그림(12)의 mux 컨트롤처럼 스테이지2에서는 곱셈계수와 곱해져야 하는 스테이지2의 a'-b'이 복소곱셈기에서 연산되도록 컨트롤을 해주었다. 이렇게 함으로써 단 한클럭의 지연도 없이 전체 FFT는 계속해서 연산이 이루어지며, 기존의 사례에서 Radix-2 MDC나 SDF 방식과 동일한 속도의 성능을 가진다. 16포인트에서 데이터의 입출력을 클럭에 맞추어 그림(17)에 나타내었다. 이것은 전체 FFT 구조에서 스테이지3과 스테이지4의 구조에서 입출력 타이밍을 보인 것이다.

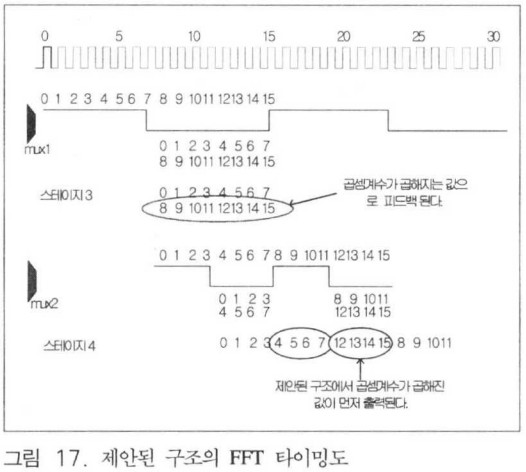


그림 17. 제안된 구조의 FFT 타이밍도

스테이지3에서 곱셈계수가 곱해지는 부분(연산점 8~15)과 스테이지4에서 곱셈계수가 곱해지는 부분(연산점 4,5,6,7,12,13,14,15)이 서로 교차되어 복소

곱셈기는 100%의 이용율로 계속 연산한다. 아래의 표(2)는 각 스테이지에서의 mux 컨트롤 신호를 나타낸 것으로 제안된 구조의 FFT 컨트롤이 매우 단순함을 알 수 있다.

표 2. 각 스테이지에서의 mux 컨트롤 신호

스테이지	mux	신호
스테이지1	mux1	'1' 32bit, '0' 32bit 반복
스테이지2	mux2	'1' 16bit, '0' 16bit 반복
스테이지3	mux3	1111111100000000 반복
스테이지4	mux4	11110000 반복
스테이지5	mux5	1100 반복
스테이지6	mux6	10 반복

제안된 구조의 64포인트 FFT 연산을 모두 마쳤을 때 출력되는 데이터 열을 표(3)에 나타내었다. 기존의 비트역순 출력 열과 조금은 다른 순서의 출력이 나오는데, 그것은 복소곱셈기의 효율을 높이기 위해서 스테이지2와 스테이지4의 출력순서에 변화를 주었기 때문이다.

표 3. 제안된 구조의 FFT 출력 데이터 열

0	32	16	8	40	24	56	12	44	28	60	4	36	20	52
2	34	18	10	42	26	58	14	46	30	62	6	38	22	54
3	35	19	11	43	27	59	15	47	31	63	7	39	23	55
1	33	17	9	41	25	57	13	45	29	61	5	37	21	53

V. 결 론

본 논문에서 제안한 하드웨어 구조에 대해 Matlab 시뮬레이션을 통해 FFT 연산된 값과 제안된 구조의 FFT의 결과값을 비교하여 검증하였다. 첫 번째 입력 값이 FFT에 입력되어 연산을 모두 마치고 출력될 때까지 총 클럭 지연은 66클럭이다. 그림(18)은 시뮬레이션에서 Matlab으로 뽑은 테스트 벡터를 넣었을 때 66번째 클럭에서부터 출력이 나오는 것을 보이고 있다.

제안된 구조로 64포인트 FFT를 설계하였을 때, 표(4)에서 보듯이 기존의 파이프라인 FFT 방식에서 Radix-2에서는 복소곱셈기를 $2(\log_4 N-1)$개, 즉 4개를 사용하였는데, 본 논문의 구조로 설계하여 복소곱셈기의 이용율을 100%로 높임으로 인하여 2개만을 사용할 수 있게 되었다. 이것은 Radix-4에서의 복소곱셈기 수와 동일하고 BF의 구조는 Radix-2이기 때문에 더욱 간단하여 성능대비 면적비가 더욱 우수하다. 또한 현재까지 발표된 논문중에서 성능대비 면적비가 가장 우수한 Radix-2^2 SDF 방식[8]과 비교할 때 동일한 속도 성능과 하드웨어 리소스를 갖는다. Radix-2^2 SDF 방식이 알고리즘에서 복소곱셈기수를 줄인 반면에 본 논문에서 제안한 구조는 아키텍쳐 레벨에서 컨트롤과 구조를 변형한 것으로 그 접근방식이 다르며 컨트롤 설계에서 매우 단순함을 알 수 있다.

한 개의 복소곱셈기는 4개의 곱셈기와 2개의 덧셈기로 이루어지므로, 본 논문에서 구현한 구조는 복소곱셈기 수를 절반으로 줄이기 때문에 하드웨어 리소스를 획기적으로 줄였다고 볼 수 있다. 현재 가장 많이 사용되고 있는 64포인트 Radix2-SDF와 비교하여 삼성 0.5um CMOS 스탠다드 셀 라이브러리[9]를 근거로 예측하여 표(5)에 나타내었다. 제안된 구조는 약 52000개의 게이트로써 R2-SDF의 약 60% 정도의 하드웨어 리소스를 가지게 됨을 볼 수 있다. 또한 더 큰 크기의 FFT에 사용함에 있어서 제안된 구조의 효과를 표(6)에나타내었다. 1k, 4k, 8k와 같이 하드웨어 리소스를 많이 갖는 FFT에서

표 4. 제안된구조와 여러가지 파이프라인 64포인트FFT 구조의 비교

구 조		BF이용율	복소곱셈기수	복소곱셈기 이용율	지연소자수	복소덧셈기수
R 2	MDC	50%	4	50%	94	12
	SDF	50%	4	50%	63	12
	제안된구조	**50%**	**2**	**100%**	**63**	**12**
R 4	MDC	25%	6	25%	156	24
	SDF	25%	2	75%	63	24
	SDC	100%	2	75%	126	18
Radix-2^2 SDF		50%	2	75%	63	12

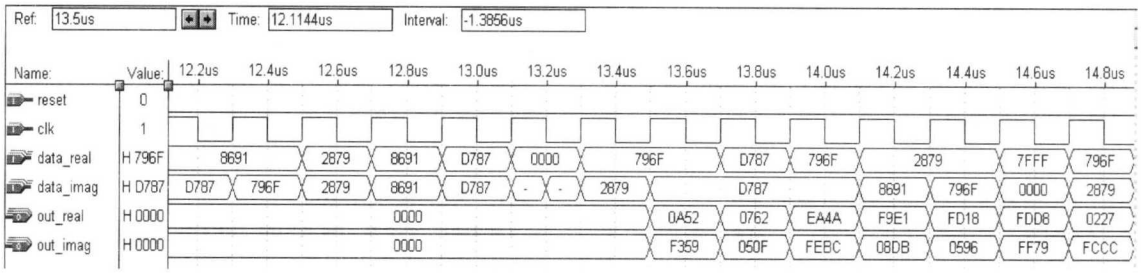


그림 18. 제안된 구조의 FFT 시뮬레이션 결과

본 논문의 제안된 구조는 더욱 효과적으로 적용 될 것이다. 앞으로 OFDM, 디지털 신호처리, xDSL, DAB, DVB 등의 전체 시스템이 하나의 칩(System On a Chip)으로 구현될 때 본 논문의 제안된 구조는 성능 대 면적 비가 가장 우수하므로 널리 활용이 가능할 것으로 기대된다. 또한 본 제안한 구조를 75%의 복소곱셈기 이용율을 가지는 Radix-4 알고리즘에 Split Radix FFT[10], Mixed Radix FFT[11] 등과 함께 그 구조를 변형 적용하여 복소곱셈의 타이밍을 두 스테이지에서 한개를 사용하도록 그 구조를 더욱 확장하게 되면 5개의 스테이지에서 2개의 복소곱셈기만을 사용할 수 있을 것으로 기대되며, 이에 대한 연구가 계속 진행중이다.

표 5. R2-SDF와 제안된 구조의 하드웨어 리소스 비교

	R2-SDF		**제안된 구조**	
	수량	게이트 수	수량	게이트 수
지연소자	63	4536	65	4680
Butterfly(BF)	6	5491	6	5491
복소곱셈기	4	67724	**2**	**33862**
기 타		7000		8000
합 계		84751		**52033**

표 6. R2-SDF와 제안된 구조의 여러 가지 크기의 FFT 비교

	R2-SDF		**제안된 구조**	
	복속곱셈기 수	BF 수	복소곱셈기 수	BF 수
256	6	8	**3**	8
1k	8	10	**4**	10
4k	10	12	**5**	12
16k	11	13	**6**	13

참 고 문 헌

[1] E.H. Wold and A.M. Despain, *"Pipeline and parallel pipeline FFT processors for VLSI inplenentation,"*, IEEE Trans. Comput C-33(5), pp414~426, 1984.

[2] R.Van Nee and R. *Prasad, OFDM for Wireless Multimedia Communicat*ion, Artech House, pp33~50, 2000.

[3] Alan V. Oppenheim and Ronald W. *Schafer, Discrete Time Signal Proces*sing, Prentice Hall, pp581~605, 1989.

[4] 김재석, 조용수, *조중휘, 이동통신용 모뎀의 VLSI 설계*, 대영사, pp322~329, 2001.

[5] L.R. Raviner and *B. Gold, Theory and Application of Digital SignalProcessing,* Prentice-Hall, Inc, 1975.

[6] A.M. *Despain, "Fourier transform computer using CORDIC iterations,"*, IEEE Trans, Comput. C-23(10), pp993~1001, 1974.

[7] G.Bi and E. *V. Jones, "A pipelined FFT processor for word-sequential* data,", IEEE Trans. Acoust. Speech, Signal Processing, 37(12), pp1982~1985, 1989.

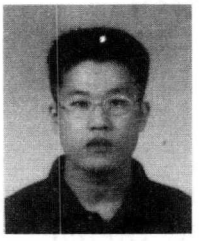
[8] S. He and M. Tork*elson, "A new approach pipeline FFT processor*," IEEE Proceedings of IPPS, pp766~770, 1996.

[9] Samsung Elect*ronics, ASIC STD85/STDM85 0.5um High Density CMOS Standard Cell Library*, September 1997.

[10] Jesus Garcia, Juan A. M, A*ngel M. B, "VLSI Configurable Delay Commutator for a Pipeline Split Radix FFT Architec*ture," IEEE Tran. Signal Processing, v 47, pp3098~3107, Nov 1999.

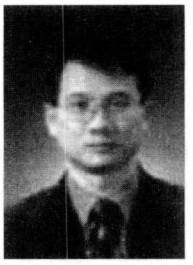
[11] Gordon L, DeMuth, "*Algorithm for defining mixed radix FFT flow* graphs," IEEE Tran. Signal Processing, v37, pp1349~1358, Sept 1989.

이 종 민(Jong-min Lee)



2001년 2월 : 광운대학교
전자공학과 졸업
2001년 3월~현재 : 광운대학교
전자공학과 석사과정
<주관심 분야> 무선통신,
신호처리

정 용 진(Yong-jin Jeong)



1983년 2월 : 서울대학교
제어계측공학과 졸업
1983년 3월~1989년 8월 :
한국전자통신연구원
1991년 5월 : 미국 UMASS
전자전산공학과 석사
1995년 2월 : 미국 UMASS
전자전산공학과 박사
1995년 4월~1999년 2월 : 삼성전자 반도체 수석연
구원
1999년 3월~현재 : 광운대학교 전자공학부 조교수
<주관심 분야> 컴퓨터 연산 알고리즘, ASIC 설계, 무
선통신, 정보보호